Amendments to the Specification:

Please replace paragraph [16] with the following amended paragraph:

[16] In another embodiment, circuitry within the instruction cache ~~220~~ 210 itself may be disabled by the HIT/MISS output from the UOP cache 240. As is known, operation of a typical cache occurs in two phases. First, a lookup operation is performed to determine if requested data is present in the cache (shown schematically as cache lookup 212). Second, if the data is present in the cache, a data fetch operation is performed (shown as cache fetch 214). Traditionally, cache lookups and data retrieval occurred as simultaneous operations. In an embodiment, cache fetch circuitry 214 within the instruction cache 210 may be disabled based on the status of the HIT/MISS output from the UOP cache 240. When the UOP cache indicates a hit, the cache fetch circuitry 214 may be disabled; when the UOP cache 240 indicates a miss, the cache fetch circuitry 214 may be enabled.

Please replace paragraph [20] with the following amended paragraph:

[20] According to an embodiment, the UOP cache 310 may include a delay path 370 between the cache lookup 350 and data fetch 360 units. This embodiment finds application in designs where power consumption holds a priority over instruction throughput. In this embodiment, decoded uops may be output to the execution unit at the same time, regardless of whether they are found in the UOP cache 310 or the instruction cache 320. If found in the UOP cache 310, a hit/miss output from the lookup unit ~~360~~ 350 may disable the instruction synchronizer 330, instruction decoder 340 and, optionally, portions of the instruction cache ~~310~~ 320 (via a connection not shown). If not, decoded uops may be provided to the execution unit from the instruction cache 320 by way of the instruction synchronizer 330 and instruction decoder 340. Regardless of the path, the decoded uops would be presented to an output multiplexer 380 at the same time.

Please replace paragraph [32] with the following amended paragraph:

[32] FIG. 6 is a block diagram illustrating an exemplary set of instructions stored in a line ~~610~~ 510 of an instruction cache (FIG. 6 (a)). In this example, a basic block of four instructions (I1-I4) is stored in the instruction cache. The beginning of the basic block need not be aligned to the first position of the cache line 510. In the example of FIG. 6 (a), the basic block begins at a 3-byte offset from the beginning of the line 510. The fourth instruction I4 is illustrated as a jump instruction. It may terminate the basic block. The cache line 510 is shown as having a width of 16 bytes.

Please replace paragraph [35] with the following amended paragraph:

[35] In an embodiment, lines within the UOP cache 520-540 may store not only the decoded uops but also administrative data representing the offset and byte length of the instructions to which they refer. Line 520 is shown with a data field 550 and a byte length field 560. The data field 550 may store data from the decoded uops. The byte length field 560 may store information representing the length of the instructions as they appear in the line 510 of the instruction cache. Offset information may be stored within the tag field 570 of a cache entry which, in an embodiment, may be merged with set information for the cache line 510. ~~FIG. 4~~ FIG. 5 also shows $Addr_{tag}$ and $Addr_{off}$ data being input to the tag comparator 450 to refer to this embodiment

Please replace paragraph [38] with the following amended paragraph:

[38] In this embodiment, with reference to FIG. 5, when an address is applied to the UOP cache, the address decoder 440 may cause the contents of the tag field (tag and offset data) to be output to the tag comparator 450. The tag comparator 450 may determine whether a match occurs between the stored values and an input address. If a match occurs in way 0 (FIG. 6 (c)), for example, the contents of the data field and the byte length field may be read from the cache entry ~~620~~ 520.

Please replace paragraph [39] with the following amended paragraph:

[39] To determine whether to continue to read data from the UOP cache, a next address may be computed from a sum of the previous address (IP) and the byte length read from line ~~620~~ 520. This address may be applied to the UOP cache and may cause a hit or a miss. In the example of FIG. 6, a hit may be registered at way 1. This process of reading data from the cache and incrementing the address based on the value of the byte length field may continue until a miss is registered. Once a miss is registered, data may be read from the instruction cache rather than the UOP cache.

Please replace paragraph [46] with the following amended paragraph:

[46] The embodiment of FIG. 8 permits a UOP cache to support access of uops in the interior of a cache line ~~800~~ 700. For example, some instruction (say, instruction I_n) in program flow may cause a jump to instruction I_2, an offset of 5 bytes from the beginning of the instruction cache line 510 (FIG. 6). As shown in the example of FIG. 8, the instruction I_n would cause a jump into the interior of line 700, provided the UOP cache can recognize that line 700 stores instruction I_2. The embodiment of FIG. 8 provides such functionality.